Reference Manual

IBM 709/7090 Programming Systems: FORTRAN Assembly Program (FAP)

IBM

Reference Manual
IBM 709/7090 Programming Systems:
FORTRAN Assembly Program (FAP)

© 1962 by International Business Machines Corporation

MAJOR REVISION (September, 1962)
This manual, form C28-6235, will obsolete and replace the following publications:

"FORTRAN Assembly Program (FAP) for the IBM 709/7090," bulletin, form J28-6098-1;

"FORTRAN Assembly Program (FAP) for the IBM 709/7090: Supplementary Information for the 32K Version," bulletin, form J28-6186-0.

Address comments regarding this publication to
Programming Systems Publications, IBM Corporation, P. O. Box 390, Poughkeepsie, N. Y.

Engineering &
Mathematical
Sciences
Library
*TK
7885.3
[illegible]
1962

PREFACE

A 709/7090 machine language program is a sequence of machine instructions in the form of binary numbers that directs the computer in the performance of a particular task. A symbolic language program is a representation of a machine language program in a form that is more convenient to the programmer. For the most part, this involves the substitution of mnemonic symbols for the binary instruction desired, such as TRA for the binary unconditional transfer instruction. Because of its close relation to machine language, the symbolic language enables the programmer to use all the facilities of the computer that would be available to him if he were to code his program directly in machine language.

The translation of a programmer's symbolic language program into a machine language program is accomplished by an assembly program, or assembler, which interprets the mnemonics and substitutes the required binary instructions.

An assembler is similar to a compiler (such as FORTRAN) in that it produces machine language programs. However, the symbolic language used with an assembler is closely related to the machine language of the computer, while the source language used with a compiler resembles a language in which problems are commonly stated, such as mathematical notation.

Compilers have several advantages over assemblers: the language used with a compiler is easier to learn and to use; the programmer using a compiler usually does not need an intimate knowledge of the internal operations of the computer; programming is faster; and the time required to produce a finished, working program is greatly reduced, since there is less chance of programmer errors, and since most errors which are made are detected by the compiler.

The assembler compensates for these disadvantages by offering the programmer a degree of flexibility not available with any present-day compiler.

FAP (FORTRAN Assembly Program) provides a compromise between the convenience of a compiler and the flexibility of an assembler. Using FAP and 709/7090 FORTRAN operating under either the IBM FORTRAN Monitor or the 7090/7094 Basic Monitor (IBSYS), a programmer may write the major part of his program in FORTRAN, using FAP subroutines where necessary to accomplish those parts of the job for which FORTRAN is not suitable. Likewise, he may write the major part of the program in FAP, using FORTRAN subroutines for certain computational and input/output operations. For those jobs which are coded entirely in symbolic language, FAP may be used to produce an "absolute" program which can be either loaded by the Monitor or used independently of the Monitor.

Although the primary objective of FAP is the translation of symbolic language programs into machine language programs, it is also used for updating a symbolic tape by changing, deleting, or adding instructions.

A FAP program must be assembled on an IBM 709, 7090, or 7094, but can produce an object program that can be executed on an IBM 704, 709, 7090, or 7094.

This manual describes the FAP language and the programming facilities it affords. The reader is assumed to be familiar with the IBM reference manual, IBM 7090 Data Processing System, form A22-6528.

CLA XABC); either will cause a proper entry into the transfer vector.

The HED Pseudo-Operation

The HED pseudo-operation is a 704 SAP pseudo-operation which has been included in the FAP language to make it easier to change 704 symbolic programs into 709/7090 symbolic programs. In FAP this pseudo-operation has been supplanted by HEAD.

The constituents of the HED pseudo-instruction are:

1. A single character or letter or digit, but not a special character, in column 1, the first column of the location field;
2. The operation code HED in the operation field; and
3. Up to nine single characters (letters or digits, but not a special character), separated by commas, in the variable field.

The effect of HED is the same as HEAD, except that the symbol in the location field is considered to be the heading character for symbols in the variable field of the following instructions, and the symbols in both the location field and the variable field of the HED pseudo-instruction are used to define symbols in the location fields of the following instructions.

A blank in column 1 of the symbolic card indicates heading by the character zero; that is, suspension of heading.

CHAPTER 13: STORAGE-ALLOCATING PSEUDO-OPERATIONS

The BES, BSS, and COMMON pseudo-operations are used to reserve blocks of core storage for data storage or for working space. The ORG and LOC pseudo-operations are used to set the program and location counters for storage assignment.

Location Counter and Program Counter

During assembly, the location counter keeps track of the next location to be assigned to an instruction; the program counter keeps track of the next location to be assigned to a symbol. The program and location counters operate in the same manner. The program counter may be set separately from the location counter by means of the LOC pseudo-operation. The ORG pseudo-operation sets both the location counter and the program counter. The load address on a binary card is taken from the location counter, whereas symbol definitions are taken from the program counter.

The BSS Pseudo-Operation

The BSS (Block Started by Symbol) pseudo-operation is used to reserve an area of core storage within a program for data storage or for working space.

The constituents of a BSS pseudo-instruction are:

1. A symbol or blanks in the location field;
2. The operation code BSS in the operation field; and
3. An absolute expression in the variable field.

The BSS pseudo-operation performs the following two functions:

1. A symbol in the location field is defined to have the value of the next location to be assigned by the assembler at the time the BSS pseudo-operation is encountered;
2. A block of consecutive storage locations is reserved; the number of locations reserved is the value of the expression in the variable field.

Thus, the BSS pseudo-operation reserves a block of storage whose length is given in the variable field, and if there is a symbol in the location field, this symbol refers to the first location of the block. The location of the block within the program is determined by the location of the BSS card within the program deck.

The BSS pseudo-operation causes an area to be skipped, not cleared; therefore, it may not be assumed that an area reserved by a BSS pseudo-operation contains zeros. Words of zero may be generated by DEC or OCT in such cases.

The effect of a BSS pseudo-instruction on the binary output of the assembler is to cause any binary words in the punch buffer to be written out, and to cause the next output to start with a new card origin. A BSS with a count of zero has no effect on the binary output.

Any symbols in the variable field of a BSS pseudo-instruction must have been previously defined. The expression in the variable field must be an absolute expression (see Previously-Defined Symbols, page 18, and Phase Relocation Error, page 18).

Figure 13-1 illustrates the manner in which BSS affects storage allocation.

```
00000   0 00004 0 00001    ALPHA IOCD      BETA,,4
C0001                       BETA BSS       4
C0005   0 00004 0 00006    GAMMA IOCD      DELTA,,4
```

Figure 13-1

The BES Pseudo-Operation

The BES (Block Ended by Symbol) pseudo-operation is used to reserve an area of core storage within a program for data storage or for working space. The constituents of the BES pseudo-instruction are:

1. A symbol or blanks in the location field;
2. The operation code BES in the operation field; and
3. An absolute expression in the variable field.

The BES pseudo-operation performs the following two functions:

1. A block of consecutive storage locations is reserved; the number of locations reserved is the value of the expression in the variable field;
2. A symbol in the location field is defined to have the value of the next location to be assigned by the assembler after the block has been reserved.

Thus, the BES pseudo-instruction reserves a block of storage whose length is given in the variable field; if there is a symbol in the location field, this symbol refers to the location after the last location in the block. The locations of the block within the program are determined by the location of the BES card within the program deck. BSS and BES have the same effect if their location fields are blank.

The BES pseudo-instruction causes an area to be skipped, not cleared; therefore, it may not be assumed that an area reserved by a BES pseudo-operation contains zeros. Words of zero may be generated by DEC or OCT in such cases.

The effect of a BES pseudo-instruction on the binary output of the assembler is to cause any binary words in the punch buffer to be written out, and to cause the next output to start with a new card origin. A BES with a count of zero has no effect on the binary output.

All symbols appearing in the variable field of a BES pseudo-operation must have been previously defined. The expression in the variable field must be an absolute expression, (see Previously-Defined Symbols, page 18, and Phase Relocation Error, page 18).

Figure 13-2 illustrates the manner in which BES affects storage allocation.

The COMMON Pseudo-Operation

The COMMON pseudo-operation is used to reserve an area of upper core storage for data storage or for working space. Typically, this pseudo-operation is used when two or more subprograms operate on the same block of information (see the discussion of FORTRAN COMMON usage, page 59).

The constituents of the COMMON pseudo-instruction are:

1. A symbol or blanks in the location field;
2. The operation code COMMON in the operation field;
3. An absolute expression in the variable field.

The COMMON pseudo-operation may be used only in a relocatable assembly. The COMMON pseudo-instruction operates in conjunction with a counter, called the common counter. This counter keeps track of the location of the next block of common storage to be assigned.

The COMMON pseudo-operation performs the following two functions:

1. A symbol in the location field is defined to be the current value of the common counter; and
2. The common counter is decreased by the value of the expression in the variable field.

Thus, the COMMON pseudo-operation reserves a block of storage in upper core storage. The length of the block is given in the variable field; if there is a symbol in the location field, this symbol is a common symbol which refers to the last location of the block (not the location after the last location, as with BES). This usage coincides with the FORTRAN rule that the name of an array refers to the logically first word of the array, since the first word is stored in the highest core storage location required by the array.

All symbols appearing in the variable field of a COMMON pseudo-instruction must have been previously defined. The expression in the variable field must be an absolute expression (see Previously-Defined Symbols, page 18, and Phase Relocation Error, page 18). If the COMMON pseudo-instruction is used in an absolute assembly, it will be flagged as an undefined operation; an invalid instruction will be generated with the prefix digit of the octal portion of the listing left blank.

The address portion of the fourth word of the program card that precedes the binary output will contain the address of the last piece of data assigned downward in common storage, that is, one more than the final contents of the common counter; this is the common break. However, if no COMMON pseudo-instructions occur in the program, this portion of the program card will be blank. In an assembly with COMMON, the common break will be listed as it appears on the program card.

```
00000  0 00000 4 00005    ALPHA PZE      BETA,4
00005                      BETA BES      4
00005  0 00000 4 00006    GAMMA PZE      DELTA,4
```

Figure 13-2

PART IV: UPDATING SYMBOLIC TAPES

CHAPTER 27: GENERAL DISCUSSION

The FAP assembler has two objectives: the first is translating symbolic language to machine language, assigning storage locations, and performing other operations necessary to produce the final object program - this is known as assembly; the second is updating a symbolic tape by changing, deleting, or adding instructions. The update may occur concurrently with the assembly of the updated symbolic language program.

Updating involves up to three tapes: one, the System Input tape which is used by all assemblies, compilations, and executions; and two tapes which are used only for updating, the Update Input tape and the Update Output tape.

The System Input tape contains all the jobs to be processed by a monitored system. A job may have many elements: compilations, assemblies, and executions. In the case of an update, the corrections to be made to a specified symbolic tape constitute a job element and are contained in a job on the System Input tape.

The Update Input tape is a blocked or unblocked symbolic tape which requires updating. All additions, changes, and deletions are made on the basis of the serialization in card columns 73-80. Thus, instructions on the Update Input tape must be properly serialized. An end of file on the Update Input tape is ignored.

The Update Output tape is an updated, blocked or unblocked, symbolic tape which includes all the requested corrections. This tape may be assembled at a later date; if it includes the necessary control cards, it may be a System Input tape.

The monitored system requires two tapes which deserve note. The System Listing tape contains the listing output for all successful elements of all jobs on the System Input tape. The System Punch tape contains row or column binary card images for the processed jobs.

Uses of the Update Facility

FAP's Update facility may be used for: maintaining the 709/7090 FORTRAN System; multi-reel input and output; subdividing of, or extracting from, an input tape; spacing tape; and sequence checking. See examples of some of these uses below (Figures 30-1 through 30-5).

Blocked Update Output Tape

If the programmer requests a blocked Update Output tape, this symbolic tape will be blocked 14 words per card, 16 cards per block. Control cards and END cards are always unblocked.

Cards with an * or $ in column 1 are considered to be control cards and are unblocked if they follow another such card, or any update pseudo-operation, or an END card. They are considered to be remarks cards and are blocked if they follow any other card.

Sequence Checking and Serialization

Card images on the System Input and Update Input tapes will be checked for proper sequencing by the serialization in card columns 73-80. If the cards are determined to be out of sequence, an error message is printed on- and off-line, and assembly continues.

The Update Output tape and the System Listing tape may be reserialized by using the NUMBER pseudo-operation (see description below). In the case of the Update Output tape, this provides for proper serialization for a subsequent assembly.

Corrections to the Update Input tape are made on the basis of the serialization in card columns 73-80 of the instructions on the System Input and Update Input tapes. A serialized instruction on the System Input tape will replace an instruction with matching serialization on the Update Input tape; an instruction with nonmatching serialization will be inserted in sequence. In the instruction serialization, a BCD blank is sequenced following the character * and preceding the character /. If columns 75-80 of a card image on the System Input or System Update tapes are all blank, the serialization is taken to be all zeros for sequencing, and the instruction is used immediately. Instructions that are inserted or replaced appear, so labeled, on the pre-processor assembly listing.

Serialization on an update pseudo-instruction causes an instruction on the Update Input tape with identical serialization to be deleted before the pseudo-operation is interpreted. Thus, there will not be an instruction with this serialization on the Update Output tape or on the System Listing tape. An update pseudo-operation may be serialized to indicate at what point the pseudo-operation should be interpreted.

Tape Positioning

Update pseudo-instructions, except SKIPTO, do not require matching serialization. To ensure proper spacing of tapes, it is necessary to have symbolic instructions on the System Input tape whose serialization matches those on the Update Input tape. These "spacer" cards will position the Update Input tape to the next instruction to be updated.

It is also necessary for the System Input tape to include a serialized END or ENDUP pseudo-instruction to properly position the Update Input tape at the end of the update. The END pseudo-instruction serialization must match that of the END pseudo-instruction on the Update Input tape.

The ENDUP pseudo-instruction is used for an update without assembly. If used, it must immediately follow an instruction (usually an END) whose serialization indicates where the Update Input tape is to be positioned. The ENDUP pseudo-instruction must be properly serialized or have no serialization.

If a multi-reel update has been terminated because of source program or machine error, the Update Input and Update Output tapes may be positioned incorrectly. Exercise caution in restarting or continuing. The Source and Machine Error Records will attempt to reposition a single input tape or single output tape correctly.

Illegible Input Instructions

All instructions on an input tape which are either redundant or in the wrong mode are considered illegible. If there are illegible instructions on the System Input tape, the assembly is terminated. If an instruction or block of instructions on the Update Input tape is illegible, the instruction(s) is omitted, a message is printed on- and off-line, and the job is continued. It is possible to insert lost instructions during a later update. The omission of such instructions will usually result in an assembly error, but the Update Input tape will be properly positioned for the next assembly, unless the illegible instruction is an END pseudo-instruction or spacer card.

Updating a FORTRAN Source Program Deck

In order to update a FORTRAN source program deck, the Update Output tape must be unblocked and assembly must be deleted. The ENDUP pseudo-operation is used to terminate the job.

Listing Update Pseudo-Instructions

Update pseudo-instructions are listed in the pre-processor assembly listing and normally do not appear on the Update Output and System Listing tapes. However, if the pseudo-instruction is in error and is flagged, it is written on these tapes.

CHAPTER 28: UPDATE PSEUDO-OPERATIONS

The UPDATE Pseudo-Operation

The UPDATE pseudo-operation is used to initiate the updating mode, to assign Update Input and Update Output tapes, and to specify whether or not blocking and assembly are required. The constituents of the UPDATE pseudo-instruction are:

1. Blanks in the location field;
2. The operation code UPDATE in the operation field;
3. Two symbolic expressions and two strings of characters, all separated by commas, in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

The first subfield of the variable field is an expression designating the logical system tape number of the Update Input tape. This subfield may be void or zero, implying that there is no Update Input tape, or it may be the logical number of the tape containing the symbolic input to be updated. If the FORTRAN Monitor System is used, tapes 1 through 8 may not be used as update tapes.

The second subfield of the variable field is an expression designating the logical tape number of the Update Output tape. This subfield may be void or zero, implying that there is no Update Output tape, or may be the logical number of the tape that is to contain the updated symbolic output.

The third subfield of the variable field is a string of characters which indicates whether or not the Update Output tape is to be blocked. If this subfield is void or zero, the Update Output tape will be blocked; otherwise, the Update Output tape will be unblocked.

The fourth subfield is a string of characters which indicates whether or not assembly is required. If this subfield is void or zero, assembly is required; otherwise, assembly is not required. The UPDATE pseudo-instruction containing this subfield must appear in the first card group. Having once appeared, neither this subfield nor its preceding comma may appear on another UPDATE card. If assembly is not required, FAP is reduced to an updating or blocking routine, or both, since no table entries are made and Pass 2 is omitted; the only instructions that are recognized are Update pseudo-instructions, END pseudo-instructions, and those instructions with an * in card column 1.

Serialization in card columns 73-80 of the UPDATE pseudo-instruction may be useful for multi-reel input and output.

The NUMBER Pseudo-Operation

The NUMBER pseudo-operation is used to reserialize columns 73-80 of the symbolic instructions that are output on the System Listing and Update Output tapes. The constituents of the NUMBER pseudo-instruction are:

1. Up to six alphameric characters in the location field;
2. The operation code NUMBER in the operation field;
3. A number less than 32768 in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

The location field and the variable field define the serialization to be inserted in the instructions on the output tapes; the alphameric characters will be left-justified in card columns 73-78, with blanks omitted, and the number in the variable field will be right-justified in card columns 75-79.

A zero is automatically supplied in column 80. The programmer must ensure that the alphameric and numeric fields do not overlap in the program. Reserialization begins with the serial indicated in the location and variable fields of the NUMBER pseudo-instruction. Regardless of the contents of the location field, if the variable field is omitted, reserialization is suspended and old serial numbers (if any), in card columns 73-80, will be maintained. In order to reserialize from zero, an explicit zero must appear in the variable field, in addition to any characters which may appear in the location field.

The DELETE Pseudo-Operation

The DELETE pseudo-operation causes the deletions of one or more instructions on the Update Input tape. Deleted cards will appear on the pre-processor assembly listing labeled as deleted. The constituents of the DELETE pseudo-instruction are:

1. Blanks in the location field;
2. The operation code DELETE in the operation field;
3. THRU, when required, in the variable field; and
4. A serialization number of eight characters in card columns 73-80.

If the variable field is blank, the instruction with matching serialization will be deleted; if the variable field contains THRU, all instructions starting at the current position of the Update Input tape, up to and including the instruction with matching serialization, will be deleted. In the latter case, if matching serialization does not exist, deletions will be made up to, but not including, the next instruction of higher serialization.

The IGNORE Pseudo-Operation

The IGNORE pseudo-operation causes the deletion of one or more instructions on the Update Input Tape. Deleted cards will not appear on the pre-processor assembly listing. The constituents of the IGNORE pseudo-instruction are:

1. Blanks in the location field;
2. The operation code IGNORE in the operation field;
3. THRU, when required, in the variable field; and
4. A serialization number of eight characters in card columns 73-80.

If the variable field is blank, the instruction with matching serialization is deleted; if the variable field contains THRU, all instructions, starting at the current position of the Update Input tape, up to and including the instruction with matching serialization, will be deleted. In the latter case, if matching serialization does not exist, deletions will be made up to, but not including, the next instruction of higher serialization.

The SKIPTO Pseudo-Operation

The SKIPTO pseudo-operation will cause the deletion of one or more instructions up to, but not including, an instruction with matching serialization. The constituents of the SKIPTO pseudo-instruction are:

1. Blanks in the location field;
2. The operation code SKIPTO in the operation field;
3. Blanks in the variable field; and
4. A serialization number of eight characters in card columns 73-80.

Instructions deleted will not appear on the pre-processor assembly listing. Instructions of higher serialization will not cause this operation to be terminated.

The ENDFIL Pseudo-Operation

The ENDFIL (End of File) pseudo-operation will cause an end of file to be written on the addressed update tape. The constituents of the ENDFIL pseudo-instruction are:

1. Blanks in the location field;
2. The operation code ENDFIL in the operation field;
3. A FAP expression or logical tape number in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

If the logical tape number in the variable field is that of the current Update Output tape, assigned by an UPDATE pseudo-instruction, the last partial block of instructions waiting to be output will be written on the Update Output tape before the end of file is written. If the variable field is blank, the end of file is written on the Update Output tape.

The REWIND Pseudo-Operation

The REWIND pseudo-operation causes the addressed update tape to be rewound. The constituents of the REWIND pseudo-instructions are:

1. Blanks in the location field;
2. The operation code REWIND in the operation field;
3. A FAP expression or logical tape number in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

If the logical tape number in the variable field is that of the current Update Output tape assigned by an UPDATE pseudo-instruction, the last partial block of instructions waiting to be output will be written on the Update Output tape before it is rewound. If the variable field is blank, the Update Output tape is rewound. If the Update Input or Update Output tape is rewound, it is logically disconnected. No update operation referring to it will be executed unless it is reassigned by a subsequent UPDATE pseudo-instruction.

The UNLOAD Pseudo-Operation

The UNLOAD pseudo-operation causes the addressed update tape to be rewound and unloaded. The constituents of the UNLOAD pseudo-instruction are:

1. Blanks in the location field;
2. The operation code UNLOAD in the operation field;
3. A FAP expression or logical tape number in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

If the logical tape number in the variable field is the current Update Output tape assigned by an UPDATE pseudo-instruction, the last partial block of instructions waiting to be output will be written on the Update Output tape before it is rewound and unloaded. If the variable field is blank, the Update Output tape is rewound and unloaded. If either the Update Input or Update Output tape is rewound and unloaded, it is logically disconnected. It may be reassigned by a subsequent UPDATE pseudo-instruction; however, the operator should be informed to ready the tape by a PRINT pseudo-instruction sufficiently in advance to avoid delaying the assembler.

The SKPFIL Pseudo-Operation

The SKPFIL (Skip File) pseudo-operation causes the addressed update tape to be spaced forward until an end of file is passed. The constituents of the SKPFIL pseudo-instruction are:

1. Blanks in the location field;
2. The operation code SKPFIL in the operation field;
3. A FAP expression or logical tape number in the variable field; and
4. Serialization, which is optional, in card column 73-80.

If the logical tape number in the variable field is the current Update Output tape assigned by an UPDATE pseudo-instruction, the last partial block of instructions waiting to be output will be written on the Update Output tape before it is spaced. If the variable field is blank, the Update Input tape is spaced forward past an end of file.

The UMC Pseudo-Operation

The UMC (Updated Macro-Instruction Cards) pseudo-operation is used to write the instructions generated by a macro-instruction on the Update Output tape and to delete macro-definitions and macro-instructions. The constituents of the UMC pseudo-instruction are:

1. Blanks in the location field;
2. The operation code UMC in the operation field;
3. Blanks in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

Initially, macro-definitions and macro-instructions are written on the Update Output tape, but macro-generated instructions are not written. The UMC pseudo-operation reverses this procedure. Successive appearances of UMC cause switching from writing to not writing macro-generated instructions and, therefore, from deleting to writing macro-definitions and macro-instructions. The programmer should exercise caution in the use of this pseudo-operation, since, under alternate modes, a macro-definition may be deleted, while a macro-instruction for the same code may remain. An IRP pseudo-instruction in a macro-generated sequence is never written on the Update Output tape.

The serial numbers associated with deleted macro-definitions will be deleted. All instructions generated by the macro-instruction will be serialized with the serial number associated with the macro-instruction.

Since remaining remote sequences (see RMT, page 50) will be expanded after the END card is written, when UMC is in effect, they must be inserted before the END card by the use of the RMT* pseudo-operation.

The ENDUP Pseudo-Operation

The ENDUP (End Update) pseudo-operation signals the termination of an update with assembly deleted. The constituents of the ENDUP pseudo-instruction are:

1. Blanks in the location field;
2. The operation code ENDUP in the operation field;
3. Blanks in the variable field; and
4. Serialization, which is optional, in card columns 73-80.

If assembly is deleted, END cards will be unblocked, but will not terminate the update. This makes it possible to update more than one program at a time. If assembly is not deleted, ENDUP will be undefined. Blank serialization on the ENDUP card will cause immediate termination of the update.

The PRINT Pseudo-Operation

The PRINT pseudo-operation first causes card columns 14-72 of the PRINT card to be printed online during the first pass over the input deck; then it causes a machine halt. The constituents of the PRINT pseudo-instruction are:

1. Blanks in the location field;
2. The operation code PRINT in the operation field; and
3. A string of alphameric characters starting in card column 14.

CHAPTER 29: OPTIONAL SERIALIZATION IN UPDATE PSEUDO-INSTRUCTIONS

Only the DELETE, IGNORE, and SKIPTO pseudo-instructions require serialization in card columns 73-80. Without serialization, they have no effect.

UPDATE, NUMBER, ENDFIL, REWIND, UNLOAD, SKPFIL, UMC, and ENDUP may have serialization in card columns 73-80. If these pseudo-instructions are serialized on the System Input tape, the following occurs:

1. Either the Update Input tape is positioned at an instruction with matching serialization or, if there is no instruction with matching serialization, the Update Input tape is positioned after the instruction of lower serialization and before the instruction of higher serialization;
2. The instruction on the Update Input tape with matching serialization is deleted; and
3. The pseudo-instruction is interpreted.

If the pseudo-instruction is not serialized, steps 1 and 2 are omitted; thus, the pseudo-instruction is interpreted at the current position of the Update Input tape.

CHAPTER 30: UPDATE EXAMPLES

Figures 30-1 through 30-5 illustrate the use of some of the update pseudo-operations.

```
UPDATE   9,10
```

Figure 30-1

The instruction in Figure 30-1 may be used to merge the correction cards that appear on the System Input tape after the UPDATE pseudo-instruction, with those on logical tape 9, the Update Input tape; then assemble.

```
UPDATE   9,10          ...                F0213750
```

Figure 30-2

The instruction in Figure 30-2 may be used to position logical tape 9, the Update Input tape, to an instruction with serialization F0213750; delete F0213750; merge the correction cards on the System Input tape with those on logical tape 9, the Update Input tape, write a blocked symbolic tape on logical tape 10, the Update Output tape; then assemble.

```
* FAP
  UPDATE    ,10,,D
  ENDFIL
* END TAPE
  ENDFIL
  REWIND
  UNLOAD    9
  ENDUP
```

Figure 30-3

The instructions in Figure 30-3 may be used to write an end of file, an END TAPE card, another end of file, rewind logical tape 10, the Update Output tape; and unload logical tape 9.

```
*       DATE      6/18/62
*       JOB IDENTIFICATION
*       PACK
*       FAP
*       PAGE TITLE CARD
        UPDATE   9,10
   F01  NUMBER   0
*       DATE     6/18/62
*       JOB IDENTIFICATION
*       PACK
*       FAP
*       PAGE TITLE CARD
        COUNT    5000
        ABS
        ***
        ***
        ***
        END
*       FAP
        UPDATE   ,10
        ENDFIL
*       END TAPE
        ENDFIL
        UNLOAD
        END
    (END OF FILE)
*       END TAPE
    (END OF FILE)
```

Figure 30-4

The instructions in Figure 30-4 may be used to obtain a blocked, serialized Update Output tape, logical tape 10, for a later assembly.

```
*       FAP
        UPDATE   9,10,,D
   F01  NUMBER   0
        CAL      ERASE                                                  F0101010
        DELETE                                                          F0101200
        NZT      SYMBL                                                  F0101461
        TRA      ERROR
   F02  NUMBER   0                                                      F0199999
        DELETE                                                          F0201730
        DELETE   THRU                                                   F0201840
        DELETE                                                          F0202040
        IGNORE   THRU                                                   F0204170
   FC3  NUMBER   0                                                      F0299999
        END                                                             F0302880
        ENDUP
```

Figure 30-5

The instructions in Figure 30-5 may be used to update and reserialize the instructions on the Update Input tape, logical tape 9; to delete assembly; and to write the updated symbolic output on logical tape 10, the Update Output tape.